Trace-Based Power State Machine Modelling

Daniel Lorenz, Vincent Ortland, Kim Grüttner
OFFIS – Institute for Information Technology, Oldenburg, Germany
daniel.lorenz@offis.de

Abstract—Due to the increasing algorithmic complexity of today's embedded systems, the consideration of extra-functional properties becomes even more important. Extra-functional properties such as timing, power consumption, and temperature need to be validated against given requirements on all abstraction levels. For timing and power consumption at RT- and gate-level, several techniques are available, but there is still a lack of methods and tools for power estimation and analysis at electronic system level (ESL) and above. In today's systems most of the hardware is not design from scratch, but bought as black-box components from IP vendors. Our Power State Machine (PSM) model enables power simulation of these components at ESL by deriving the power of communication at the component's interfaces. In this work we present an Eclipse plug-in which supports the designer or user of a black-box IP component in creating the PSM model based on gate-level simulations and power estimations.

I. INTRODUCTION

There is a high demand for mobile embedded applications with increasing complexity and performance, as well as long runtimes, which leads to increasing development time and costs. At the same time, limited battery capacitance reduces runtime of high performance mobile devices. To tackle these challenges, the system must be simulated as a whole, including timing and power properties. Furthermore, it is necessary to be able to analyse the expected power consumption of all system components for realistic system use-cases, e.g. in system-level simulation runs.

This includes the consideration of energy consumption of the processor cores and its integrated power management capabilities, plus the energy consumption of additional hardware components on the platform, e.g. buses, memories, dedicated hardware accelerators. Most of them are not developed from scratch, but reused from previous designs or bought from IP vendors. Usually, these IP components do not contain any information about their power consumption. In some cases data sheets are available that allow calculating an estimate of the power consumption in dependence of parameters such as clock frequency, supply voltage, and target technology. However, information of these data sheets can hardly be used in functional system-level simulations, as they do not consider any dynamic effects such as data and state dependencies.

To enable power management evaluation of the entire system on high abstraction levels, a holistic approach for energy estimation using virtual platforms on electronic system level (ESL) is required [1].

II. PSM MODEL

Figure 1 gives an overview of our I/O observation-based approach for annotating energy information at black-box IP simulation models. The ports of the IP component are observed over time to approximate the internal functionality. Based on these observations, a Protocol State Machine (PrSM) is controlled. The main task of the PrSM is to trigger state transitions in the Power State Machine (PSM) based on the observation and interpretation of the interaction between component and environment. The state of the PSM represents the average switched capacitance of the IP component. By using the capacitance instead of power the output of the PSM is independent of Dynamic Voltage and Frequency Scaling (DVFS) parameters such as supply voltage and clock frequency. By modelling the interdependencies between I/O and internal states the PrSM extracts the energetic relevant events to orthogonalise the communication and functional artefacts of the non-functional PSM model. This may lead to a reduction of complexity in the PSM because it only describes the different internal operation modes, whereas the PrSM covers the access protocol of the component. Furthermore, the separation of PrSM and PSM has the advantage that components with the same access protocol and different internal implementations could use the same PrSM, only the PSM has to be changed.

PSM and PrSM are each modelled as an Extended Finite State Machine (EFSM) which allows the extension of a simple FSM with (shared) state variables to reduce the complexity [2].

Many designs have a data-dependent power consumption. This is in most cases caused by switching activity which strongly depends on the switching activity at the inputs. We developed an extension for the PSM model which models the data-dependent power consumption with a dynamic output per state in the PSM instead of a constant output. This extension also considers pipeline stages which may contribute differently to the whole power consumption [3].

III. PSM ECLIPSE PLUGIN

As power traces of gate-level simulations and IP component protocol descriptions serve as only input for PSM model generation, we developed an Eclipse plug-in for supporting the generation of PSM models. This plug-in can be used by the designer of an IP component for a systematic power character-
isation and state-based power model abstraction. Furthermore, the plug-in can be used to share the PSM model together with its characterisation data for future use and refinement. The inputs required by the plug-in are reference gate-level power and I/O traces for specific use-cases. Based on these data, the PSM and PrSM, including shared state variables, are developed in an incremental process. Initially a PSM, a PrSM, and optionally state variables are created. Based on these, a system-level power trace can be generated. This trace can be compared against (i.e. overlaid with) the gate-level power trace. By further correlating the I/O traces with the gate-level power trace, the state machines and state variables can be further improved and refined until the desired model accuracy has been reached.

As shown in Figure 2 the plug-in has five windows. The Package Explorer window handles all projects including the model files in XML format and input traces in VCD format, representing the individual use cases. The Power Simulation Editor shows the gate-level and system-level power traces, which are graphically overlaid for comparison reasons. Furthermore, this window shows selected I/O traces with the time axis of their corresponding power traces. This way, I/O and power traces can be directly related and correlated. The user interfaces allows for scaling and zooming the traces as well as enable/disable individual traces to be able to look into detail for defined period. In the Protocol State Machine and Power State Machine window the PrSM and PSM models are built. They show all states and the states transitions. Furthermore, each transition shows the triggered/activating event, used for synchronisation between PrSM and PSM. The Properties windows shows all information of the state machines, such as states, transitions, guards, outputs, updates on the state variables, events, and state variables, as well as the I/O traces (wires), the power traces, and an error value between system-level and gate-level power trace. The user can select a special instant of time in the Power Simulation Editor. The Properties window then updates all values relating to this timestamp. This way the user can exactly trace the value changes in the model. The outputs and updates are user written functions to enable complex operations operating on the shared state variables.

IV. CONCLUSION

In this work we briefly presented an Eclipse plug-in supporting the creation of PSM models. The model is created in an incremental process enabling arbitrary refinement of the model fidelity. The resulting model is a state- and data-dependent power model triggered by the communication at the interfaces of the examined component.

ACKNOWLEDGMENT

This work has been supported by the EnerSave Project, funded by the German Federal Ministry of Research and Education (BMBF) under Grant Agreement 16BE1102.

REFERENCES

